COMP 308
ARTIFICIAL INTELLIGENCE
PART 4 – GAME PLAYING

Njeri Ireri

Jan – April 2020

We Shall Discuss

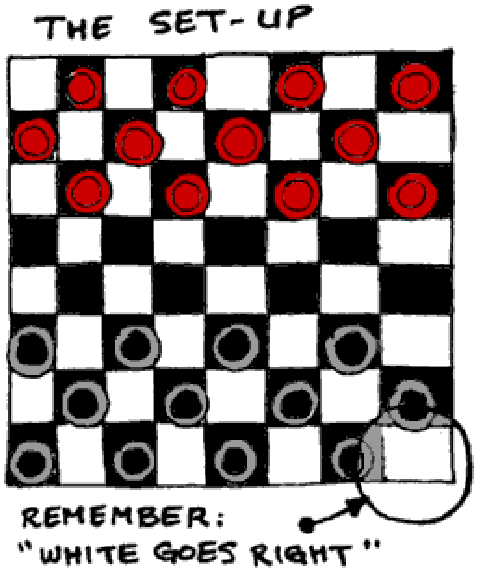
- Games. Why?
- Minimax search
- Alpha-beta pruning

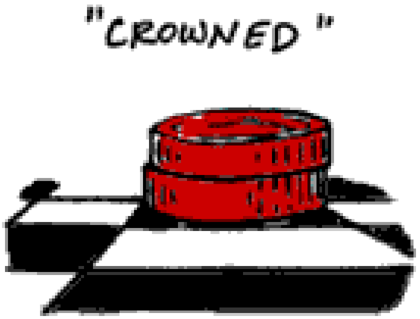
Why are people interested in games?

Some games:

- Ball games
- Card games
- Board games
- Computer games
- …

Why are people interested in games? Draughts





Is a computer more intelligent
if it beats you in a game of draughts?

Why are people interested in games? Robocup

Why study board games ?

- One of the oldest sub-fields of AI.
- Game Playing has been studied for a long time
  - Babbage (tic-tac-toe)
  - Turing (chess)
- It is an abstract and pure form of competition that seems to require intelligence
- Easy to represent the states and actions
  - Very little world knowledge required !
- Game playing research has contributed ideas on how to make the best use of time to reach good decisions, when reaching optimal decisions is impossible. These ideas are applicable in tackling real-world search problems
  - "A chess (or draughts) playing computer would be proof of a machine doing something thought to require intelligence"

Game playing

- Game playing is a special case of a search problem, with some new requirements

- Up till now we have assumed the situation is not going to change whilst we search
  - – static environment but
- Game playing is not like this, because
  - The opponent introduces uncertainty
  - The opponent also wants to win

Our limit here: 2-person games, with no chance

Game playing

- "Contingency" problem:
  - We don't know the opponents move !
- The size of the search space:
  - Chess : ~15 moves possible per state, 80 ply
    - 15^{80} nodes in tree
  - Go : ~200 moves per state, 300 ply
    - 200^{300} nodes in tree
- Game playing algorithms: they involve
  - Search tree only up to some depth bound
  - Use an evaluation function at the depth bound
  - Propagate the evaluation upwards in the tree

Types of games

	Deterministic	Chance
Perfect information	Chess, draughts, Go, Othello, Tic-tac-toe, Ajua	Backgammon, Monopoly
Imperfect information		Bridge, Poker, Scrabble

Game Playing - Chess

- Shannon - March 9th 1949 - New York
- Size of search space (10^{120} - average of 40 moves)

- 200 million positions/second = 10^{100} years to evaluate all possible games

- Searching to depth = 40, at one node per microsecond it would take 10^{90} years to make its first move

Game Playing - Chess

- 1957 - Newell and Simon predicted that a computer would be chess champion within ten years
- Simon : “I was a little far-sighted with chess, but there was no way to do it with machines that were as slow as the ones way back then”
- 1958 - First computer to play chess was an IBM 704 - about one millionth capacity of deep blue.
- 1967 : Mac Hack competed successfully in human tournaments
- 1983 : “Belle” obtained expert status from the United States Chess Federation
- Mid 80’s : Scientists at Carnegie Mellon University started work on what was to become Deep Blue.
- Project moved to IBM in 1989

Game Playing - Chess

- May 11th 1997, Gary Kasparov lost a six match game to Deep blue
  - 3.5 to 2.5
  - Two wins for deep blue, one win for Kasparov and three draws

Game Playing - Checkers (Draughts)

- Arthur Samuel - 1952
- Written for an IBM 701
- 1954 - Re-wrote for an IBM 704
  - 10,000 words of main memory
- Added a learning mechanism that learnt its own evaluation function
- Learnt the evaluation function by playing against itself
- After a few days it could beat its creator
- …. And compete on equal terms with strong human players

- Jonathon Schaeffer - 1996

- Developed Chinook

Game Playing - Checkers (Draughts)

- Chinook
- Uses Alpha-Beta search
- Plays a perfect end game by means of a database

- In 1992 Chinook won the US Open
- ….. And challenged for the world championship
- Dr Marion Tinsley, had been world championship for over 40 years
- … only losing three games in all that time
- Against Chinook she suffered her fourth and fifth defeat
- ….. But ultimately won 21.5 to 18.5

Game Playing - Checkers (Draughts)

- In August 1994 there was a re-match but Marion Tinsley withdrew for health reasons
- Chinook became the official world champion
- Schaeffer claimed Chinook was rated at 2814
- The best human players are rated at 2632 and 2625
- Chinook did not include any learning mechanism

Game Playing - Checkers (Draughts)

- Kumar - 2000
- "Learnt" how to play a good game of checkers
- The program used a *population* of games with the best competing for *survival*
- Learning was done using a *neural network* with the synapses being changed by an *evolutionary strategy*
- The best program beat a commercial application 6-0

- The program was presented at CEC 2000 (San Diego) and remain undefeated

Game Playing - Minimax

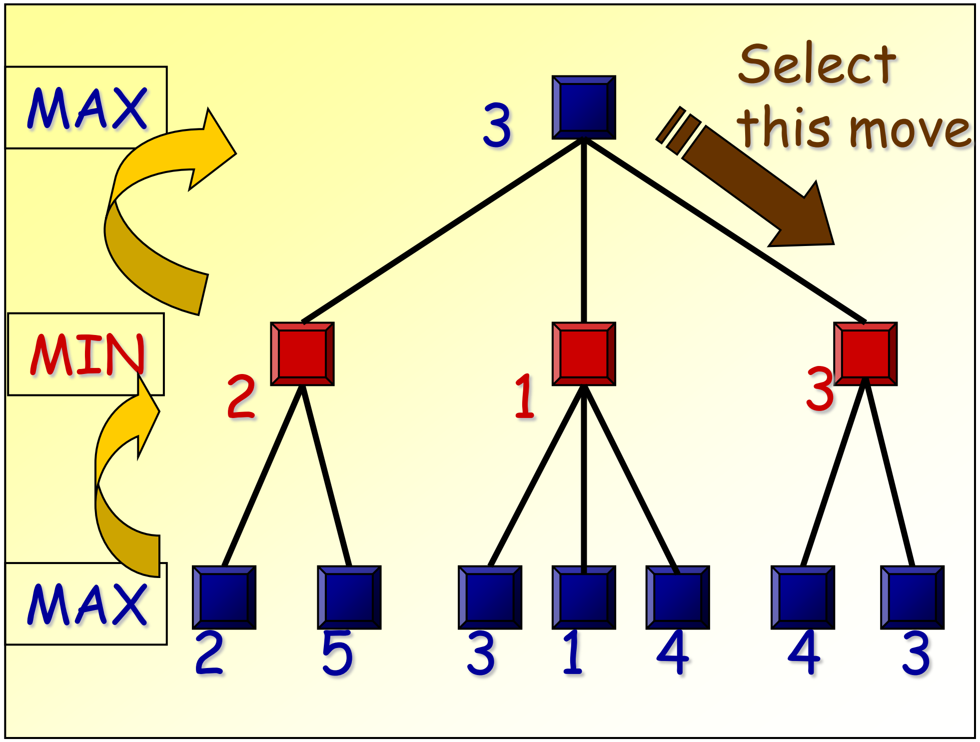
- Game Playing: An opponent tries to thwart your every move
- 1944 - John von Neumann outlined a search method (Minimax) that maximised your position whilst minimising your opponents
- In order to implement we need a method of measuring how good a position is
  - Often called a utility function (or payoff function)
    - e.g. outcome of a game; win 1, loss -1, draw 0
- Initially this will be a value that describes our position exactly

Game Playing - Minimax

- The MiniMax algorithm selects the "best" next move for a computer player in a two-player game. The algorithm makes a tree of all possible moves for both players
- This algorithm is called MiniMax simply because the computer makes moves that bring it **maximum gain**, while assuming the opponent makes moves that brings the computer **minimum gain**. Because the players alternate moves, the algorithm alternates between minimizing and maximizing levels of the recursive search tree
- Let's look at a hypothetical search tree to see how the MiniMax analysis works to ultimately select the best move; this example shows a game where there are either two or three possible moves, and where the look ahead limit (search depth) is two moves…

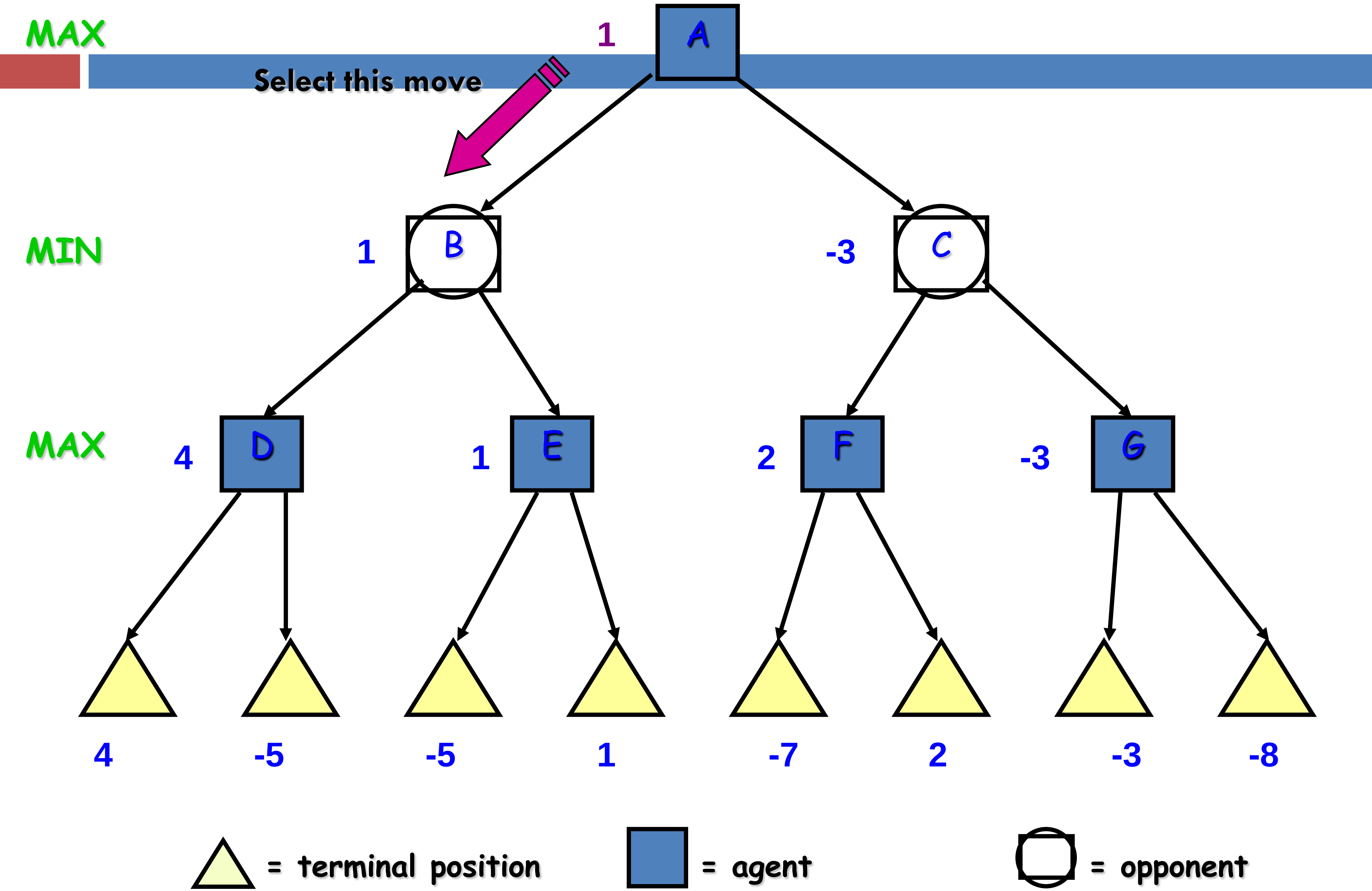
Game Playing - Minimax

- Restrictions:
  - 2 players: MAX (computer) and MIN (opponent)
  - deterministic, perfect information
- Select a depth-bound (say: 2) and evaluation function

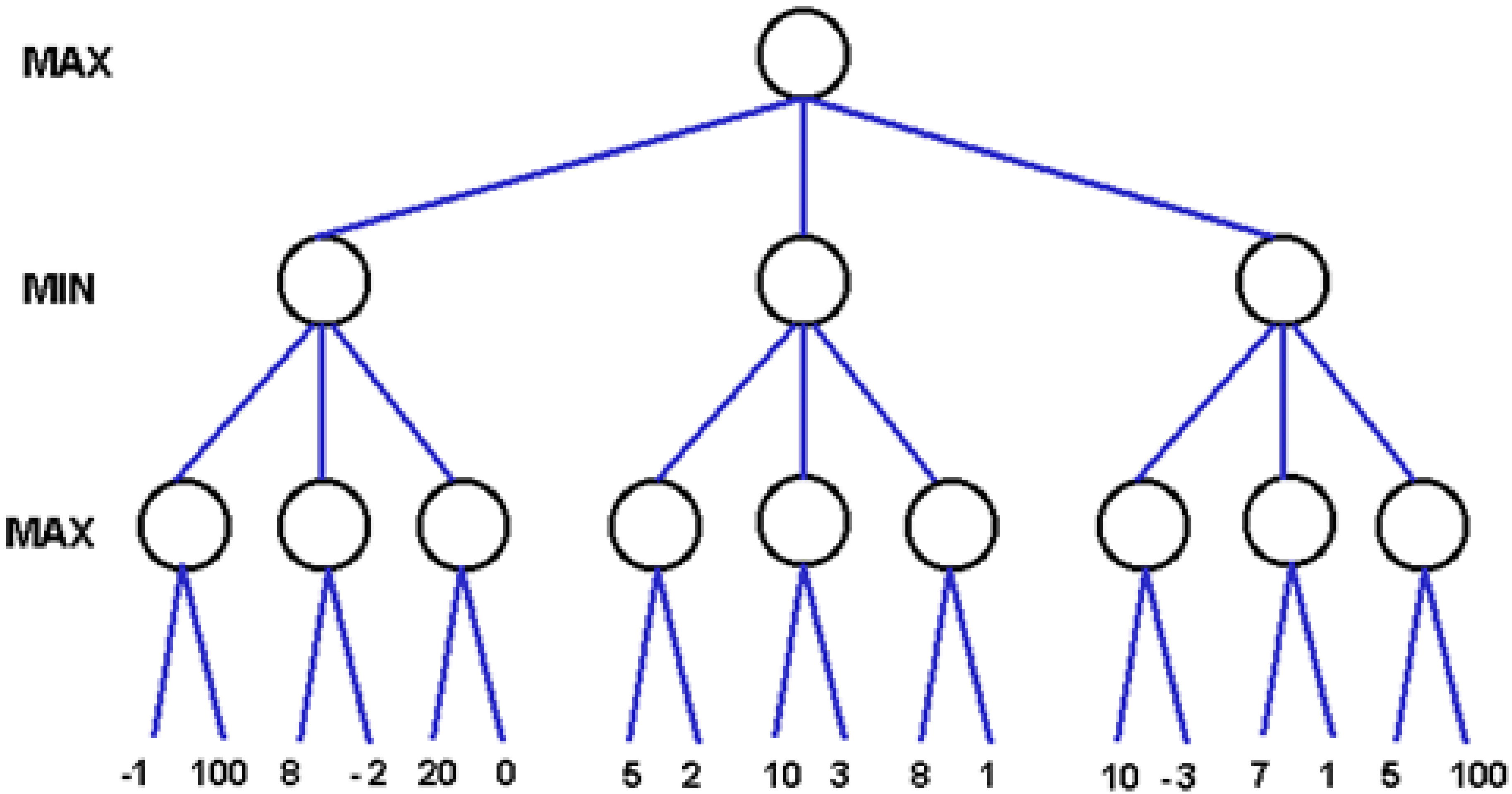


- Construct the tree up till the depth-bound
- Compute the evaluation function for the leaves
- Propagate the evaluation function upwards:
  - taking minima in MIN
  - taking maxima in MAX

Game Playing - Minimax Example



Exercise

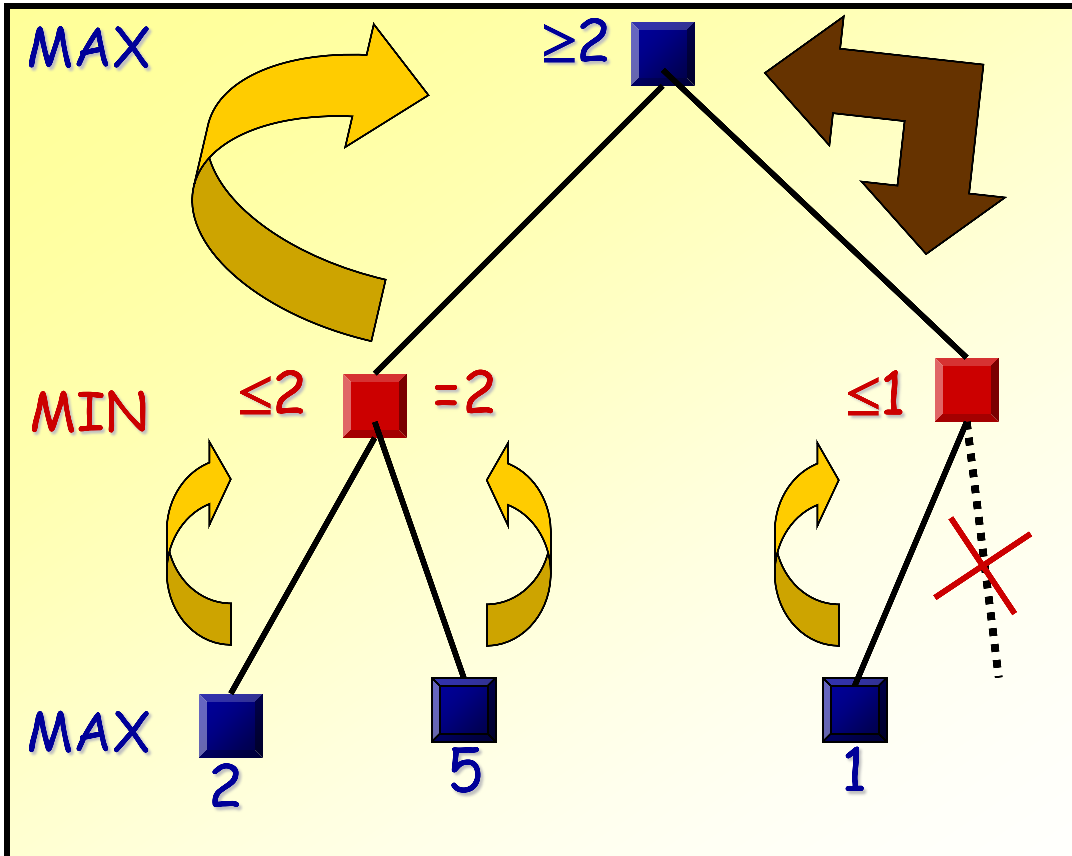


Game Playing - Alpha-Beta Pruning

- Generally applied optimization on Mini-max
- Instead of:
  - first creating the entire tree (up to depth-level)
  - then doing all propagation
- Interleave the generation of the tree and the propagation of values
- Point:
  - some of the obtained values in the tree will provide information that other (non-generated) parts are redundant and do not need to be generated

Alpha-Beta idea:

- Principles:
  - generate the tree depth-first, left-to-right
  - propagate final values of nodes as initial estimates for their parent node



- The MIN-value (1) is already smaller than the MAX-value of the parent (2)

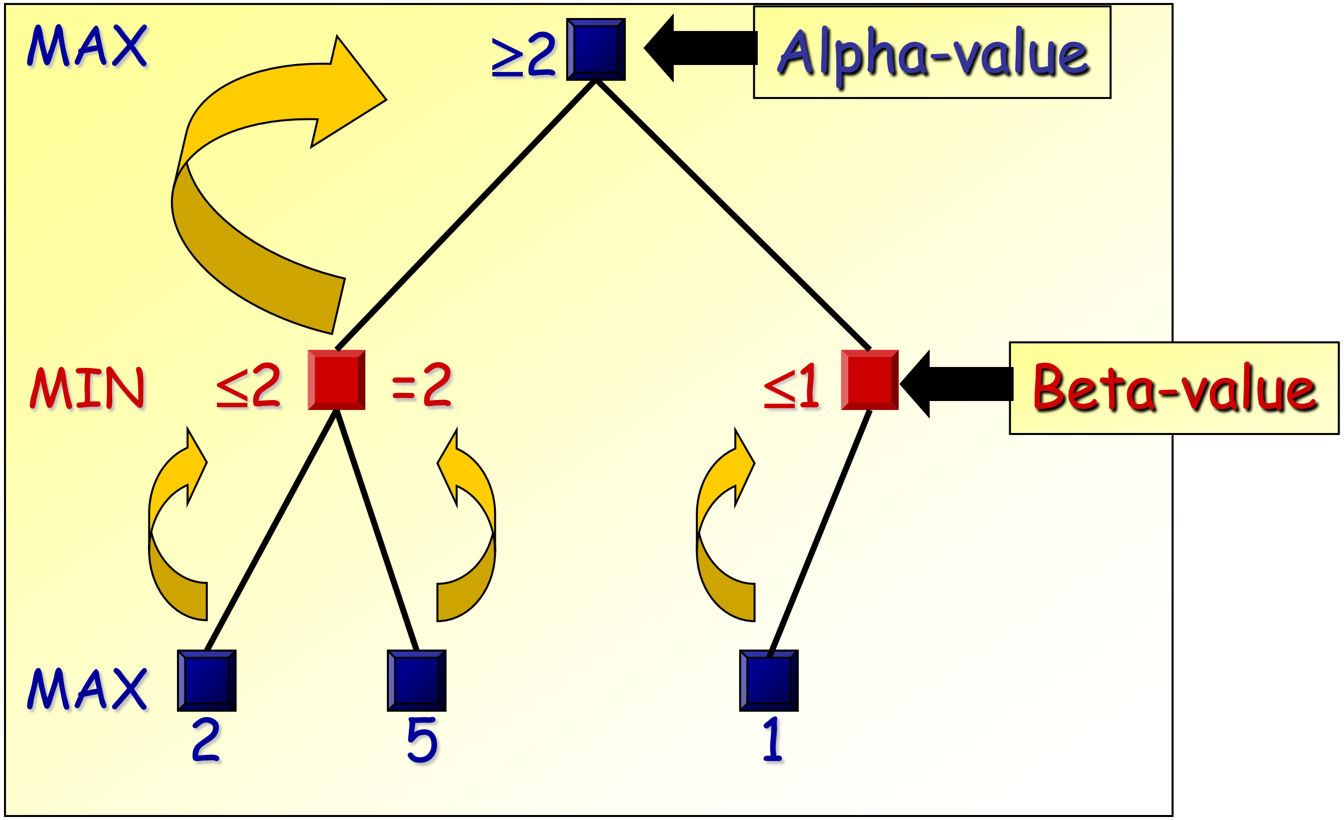
- The MIN-value can only decrease further,

- The MAX-value is only allowed to increase,

- No point in computing further below this node

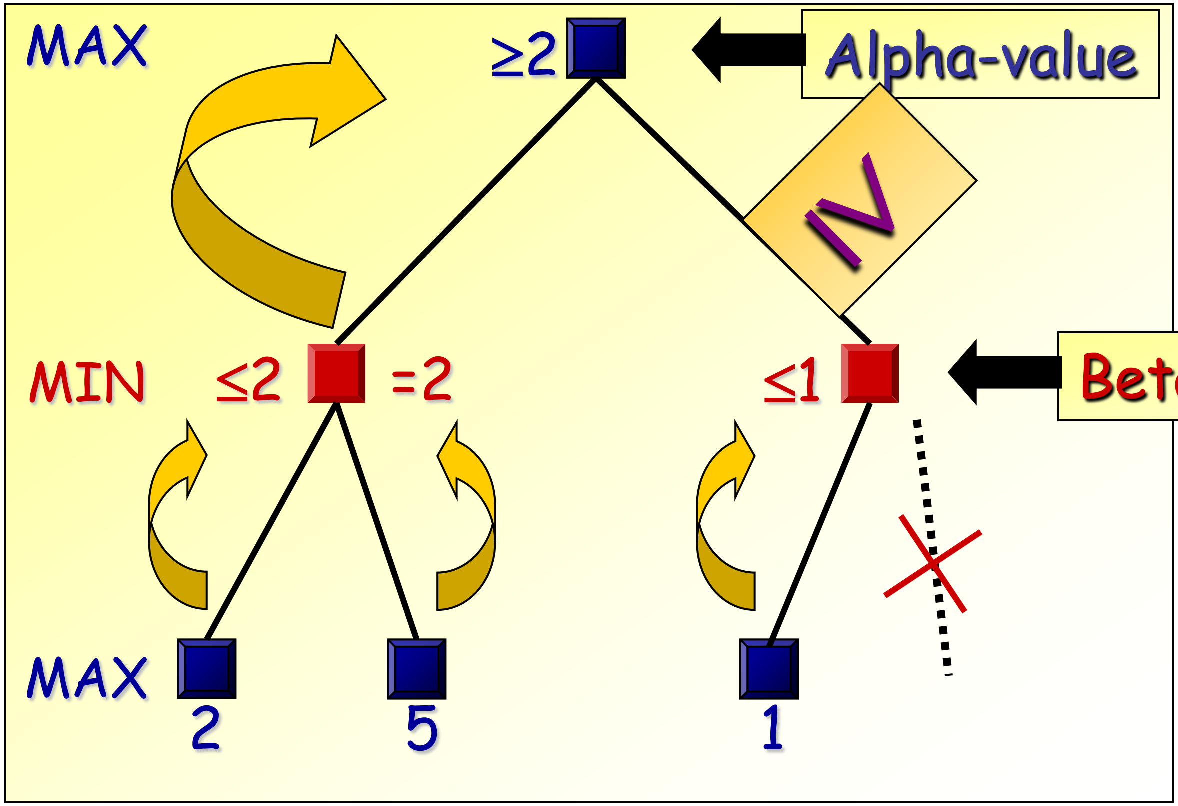
Terminology:

- The (temporary) values at MAX-nodes are ALPHA-values
- The (temporary) values at MIN-nodes are BETA-values



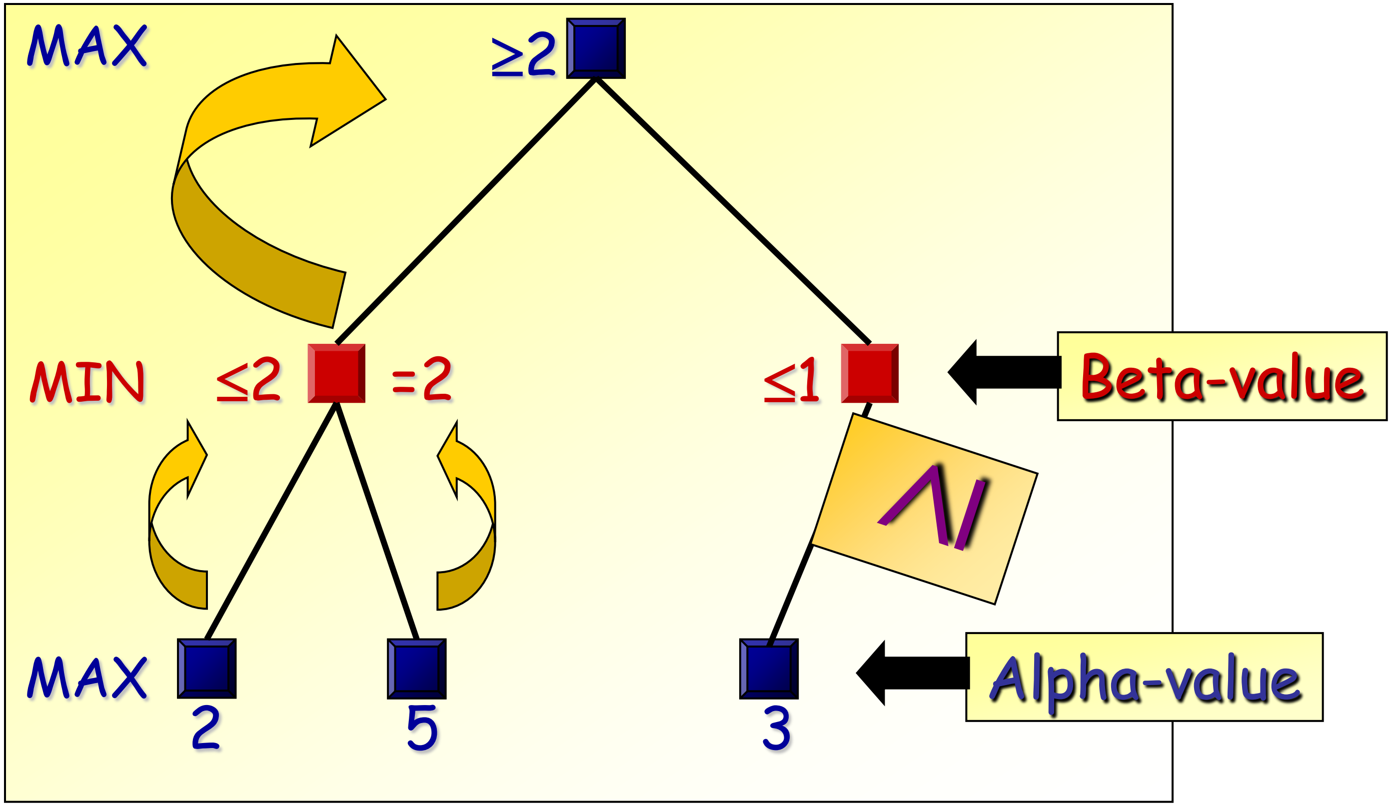
The Alpha-Beta principles (1):

- If an ALPHA-value is larger or equal than the Beta-value of a descendant node:
  stop generation of the children of the descendant

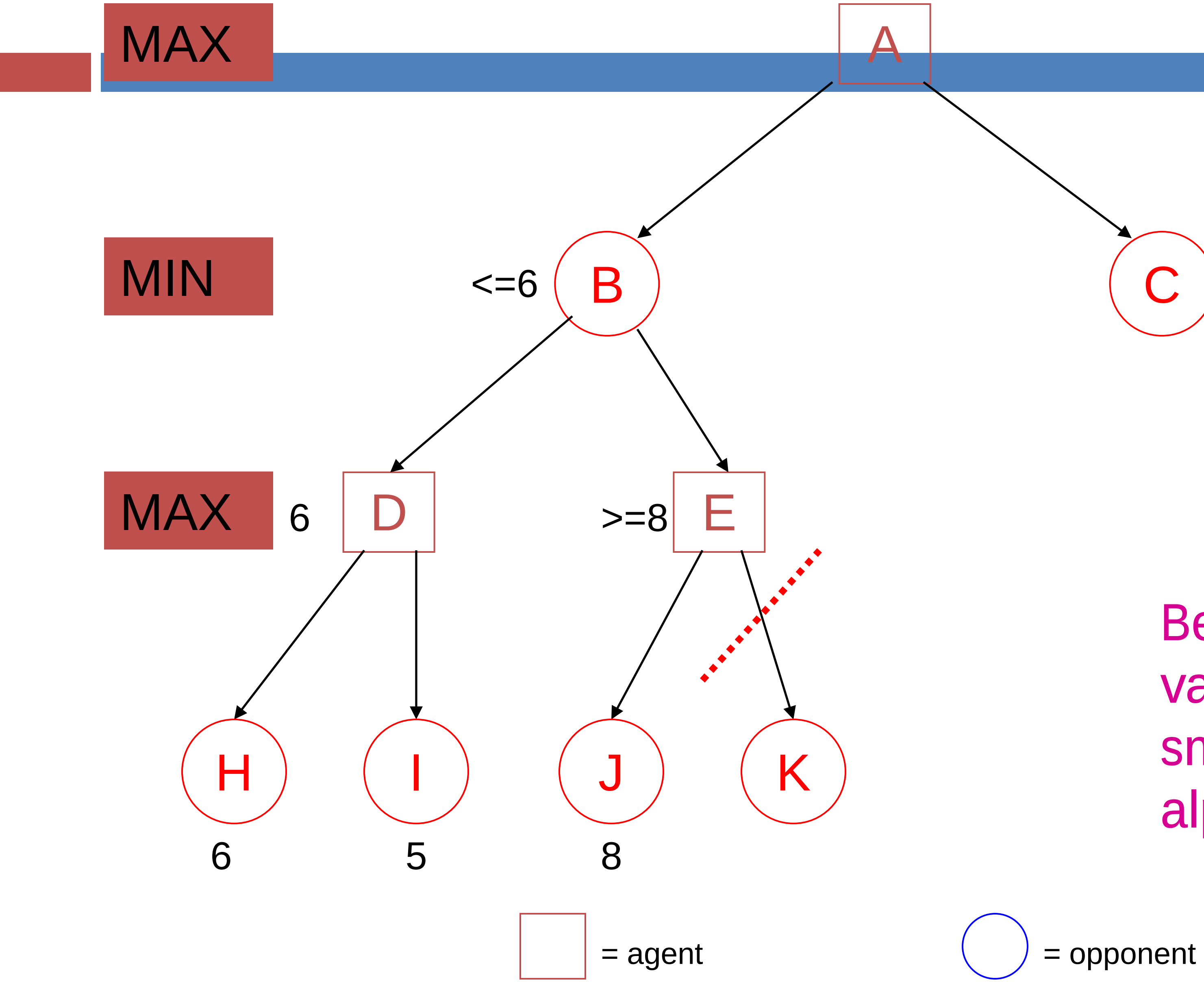


The Alpha-Beta principles (2):

- If an Beta-value is smaller or equal than the Alpha-value of a descendant node:
  <u>stop generation of the children of the descendant</u>

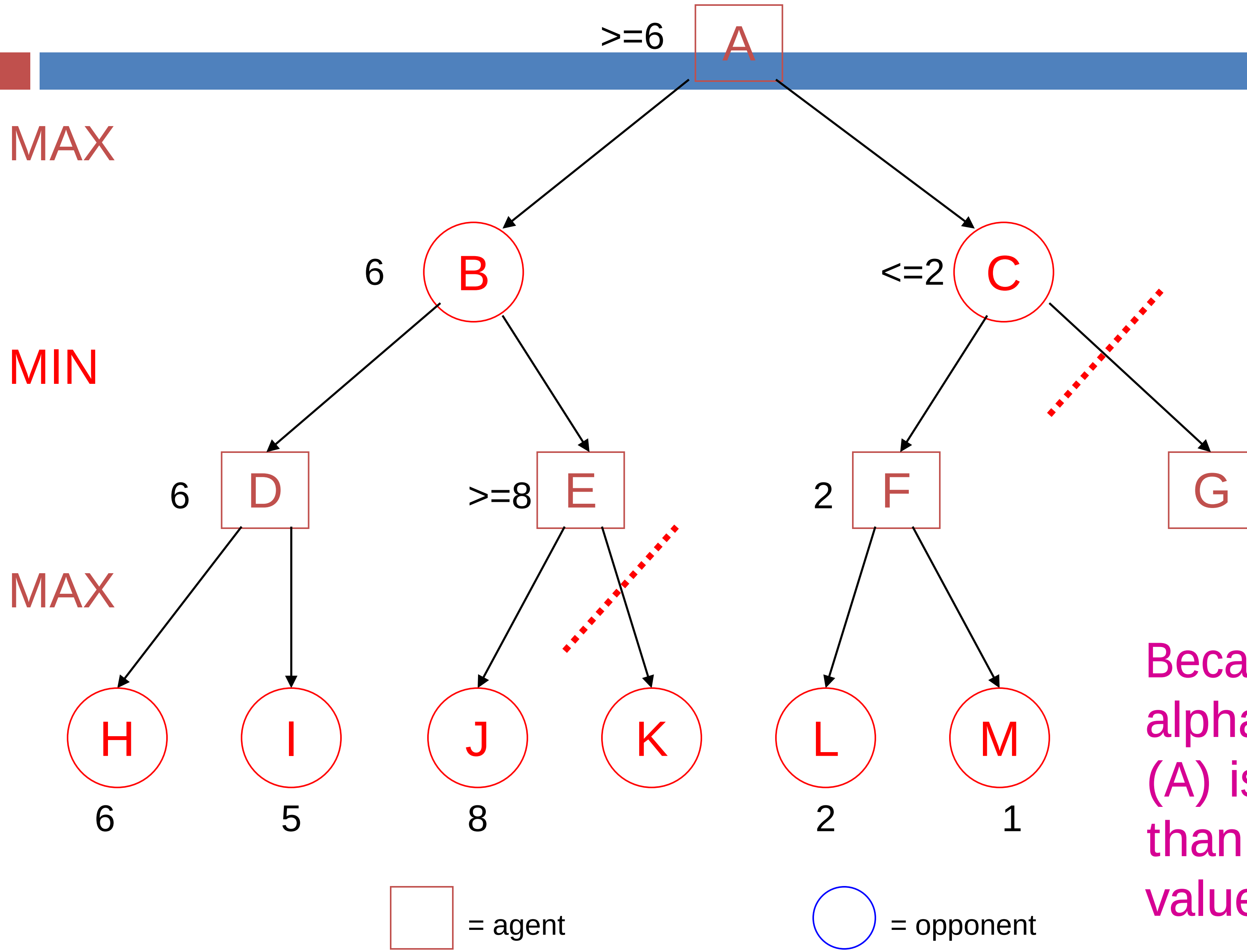


Alpha-Beta Pruning Example



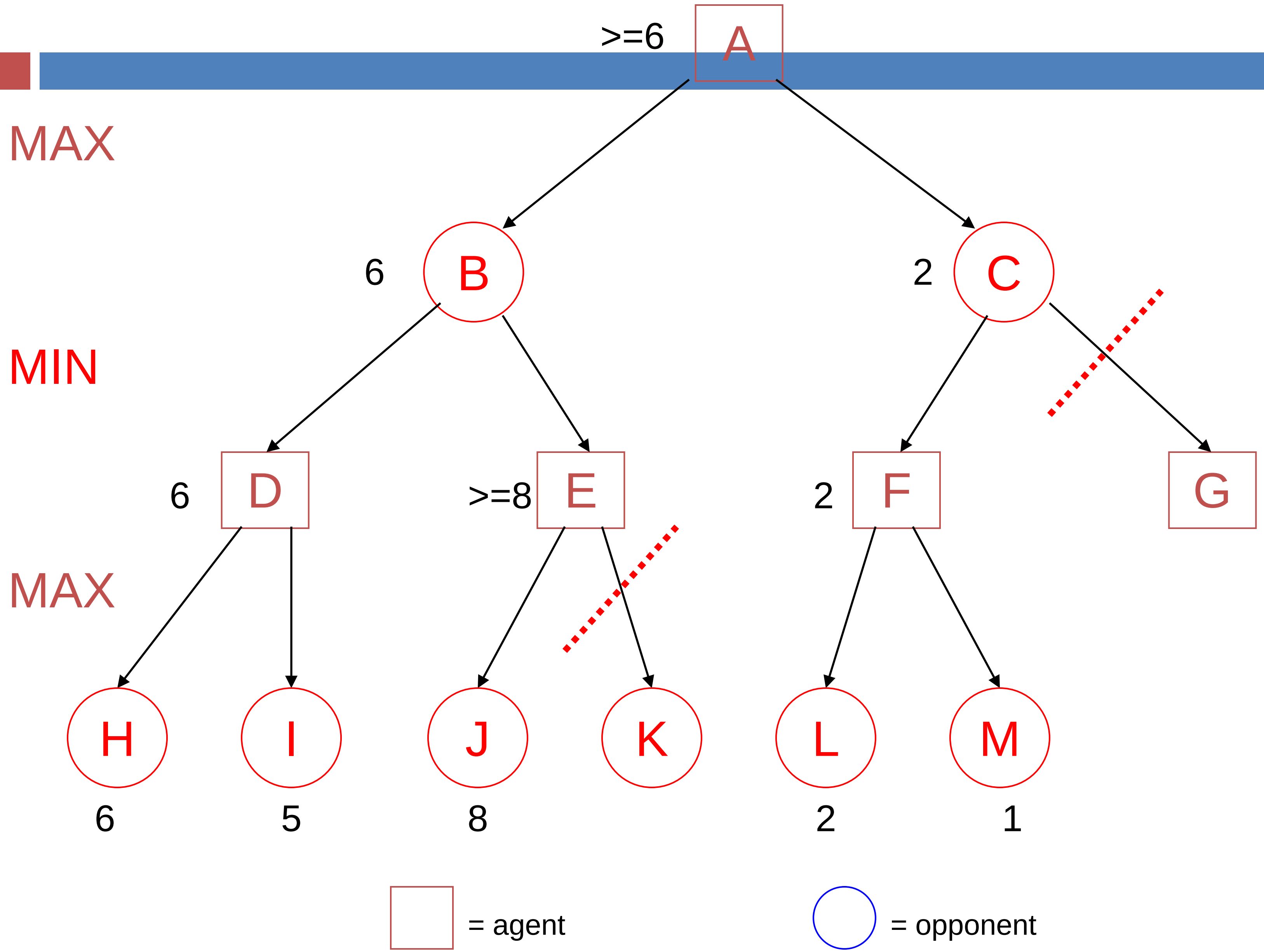
Because beta value (B) is smaller than alpha value (E)

Alpha-Beta Pruning Example

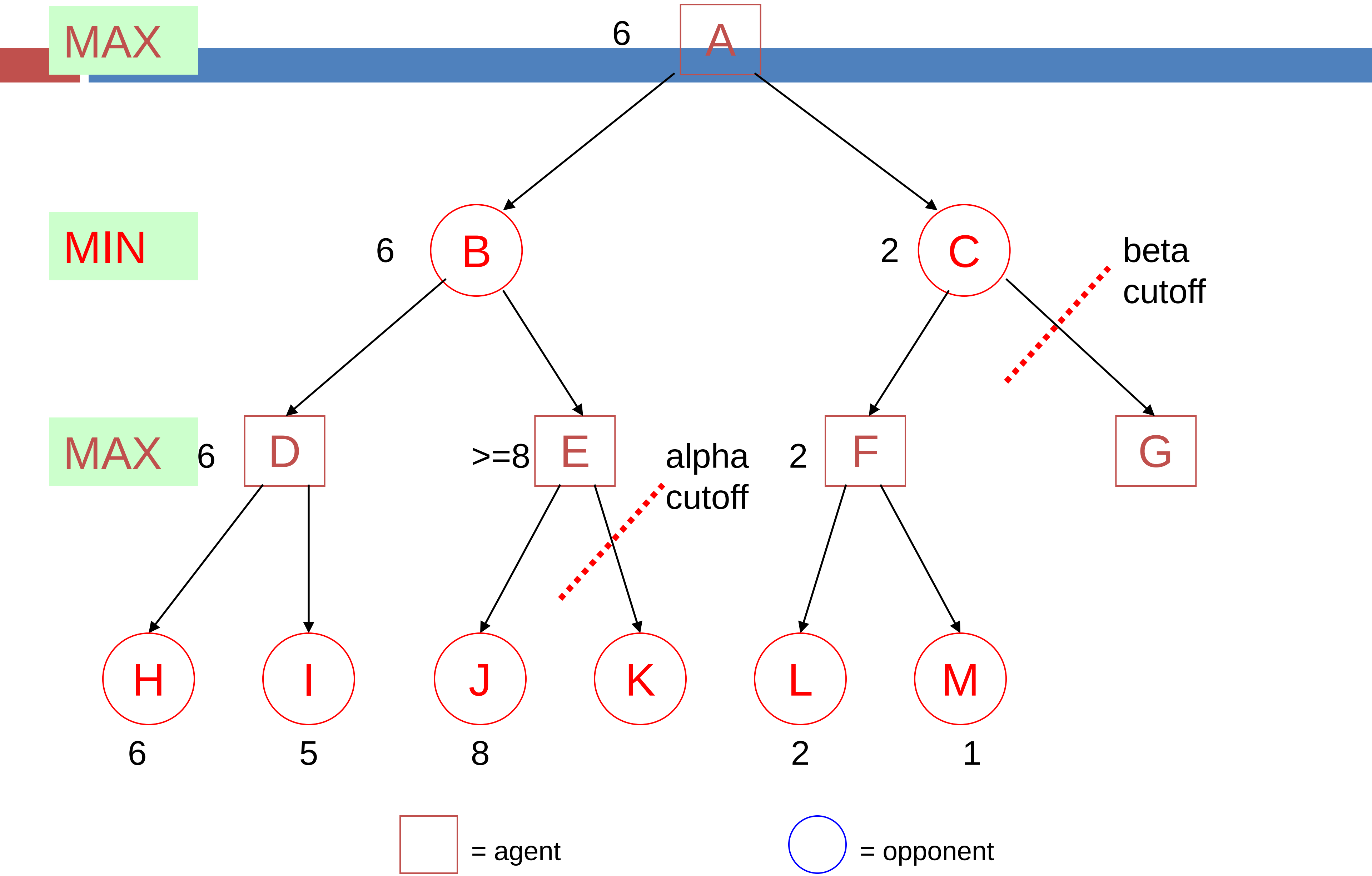


Because alpha value (A) is larger than beta value (C)

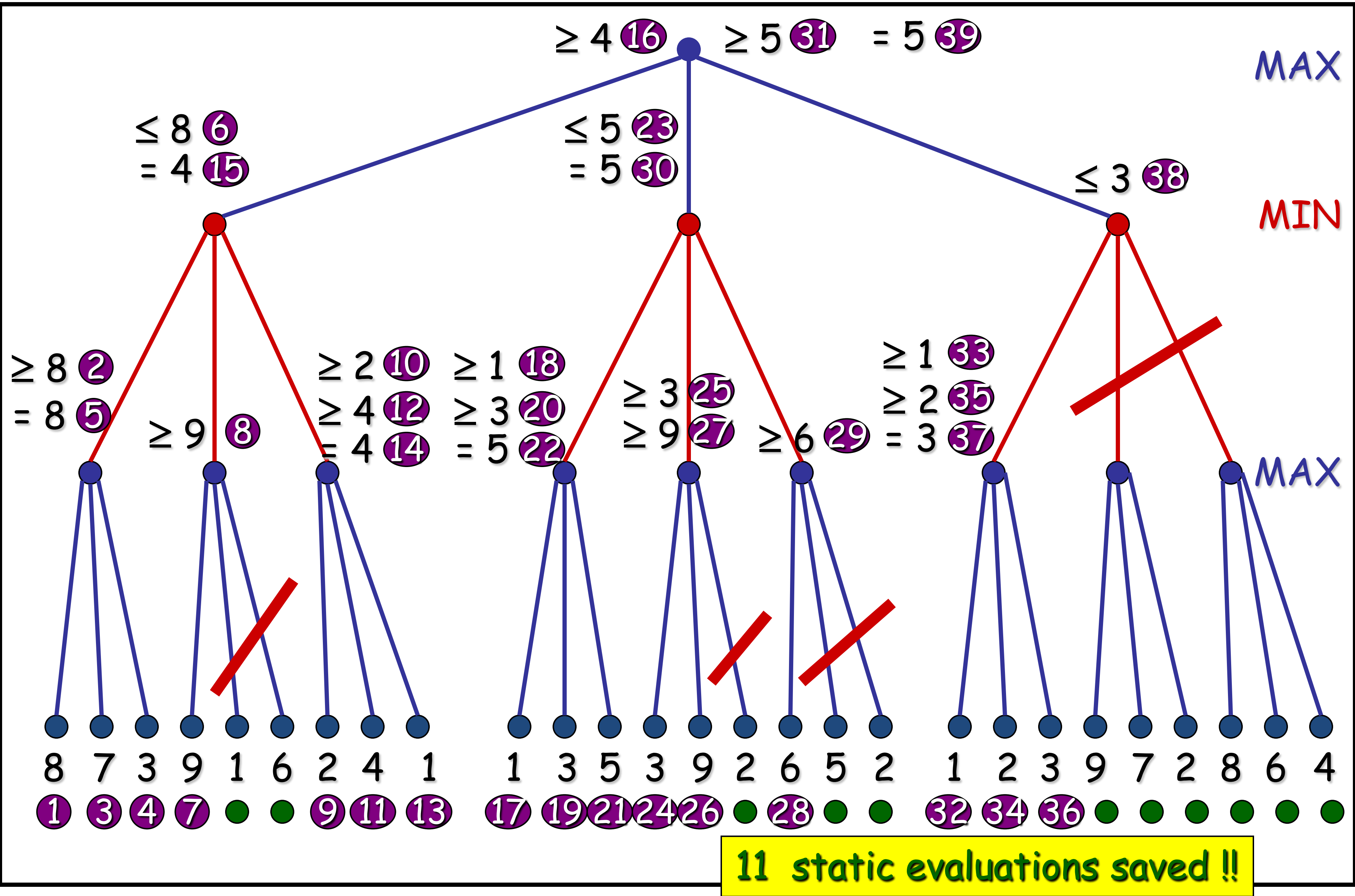
Alpha-Beta Pruning Example



Alpha-Beta Pruning Example

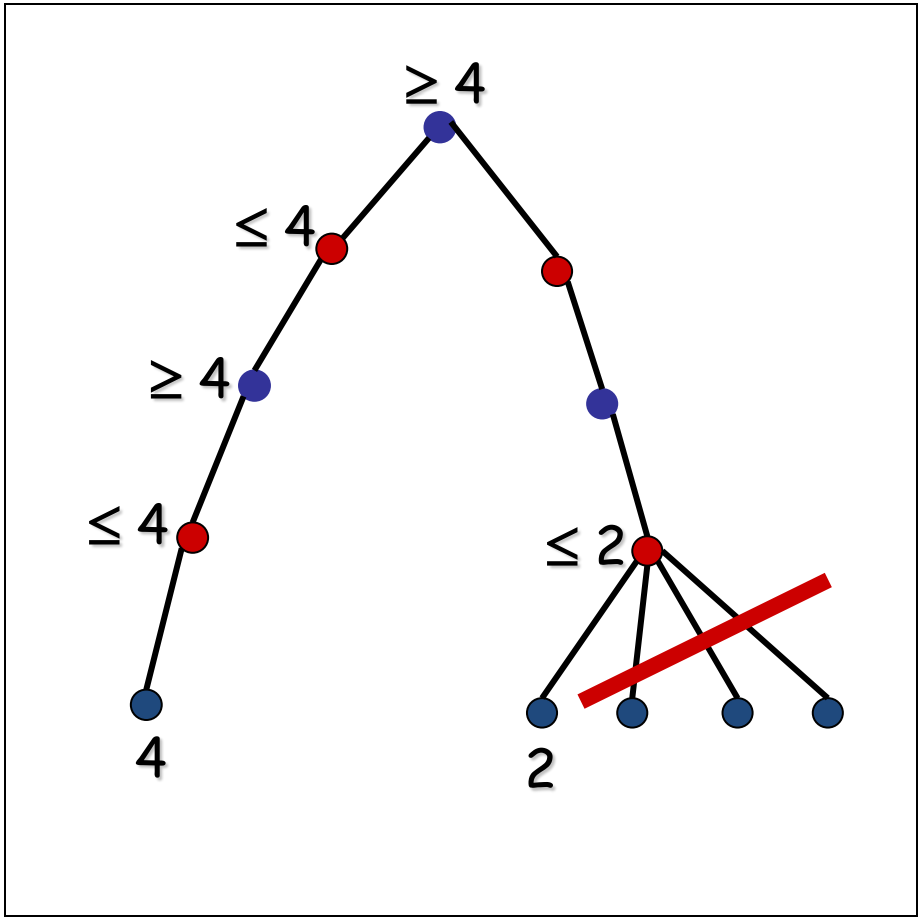


Mini-Max with $\alpha-\beta$ at work:



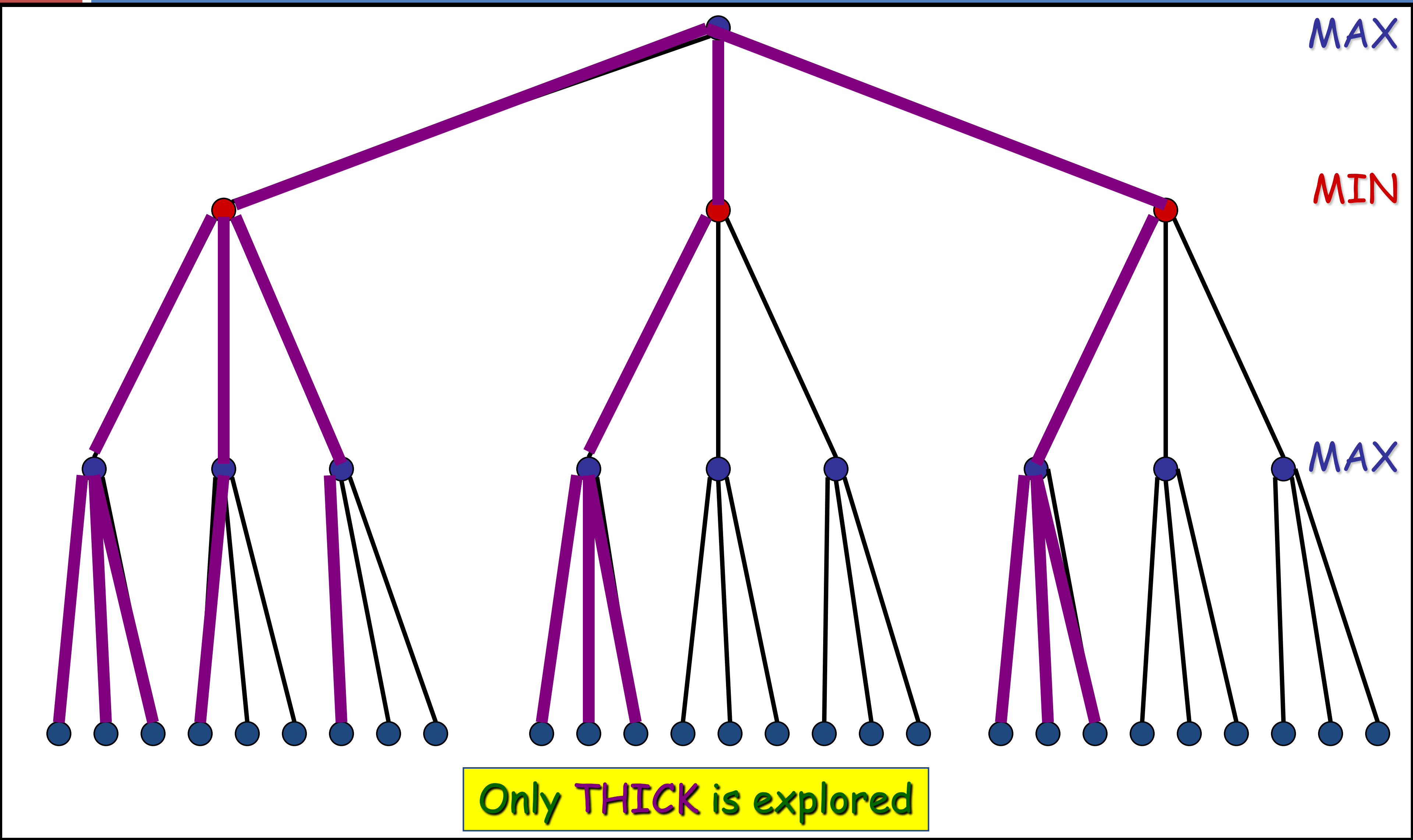
"DEEP" cut-offs

- For game trees with at least 4 Min/Max layers:
  the Alpha - Beta rules apply also to deeper levels.

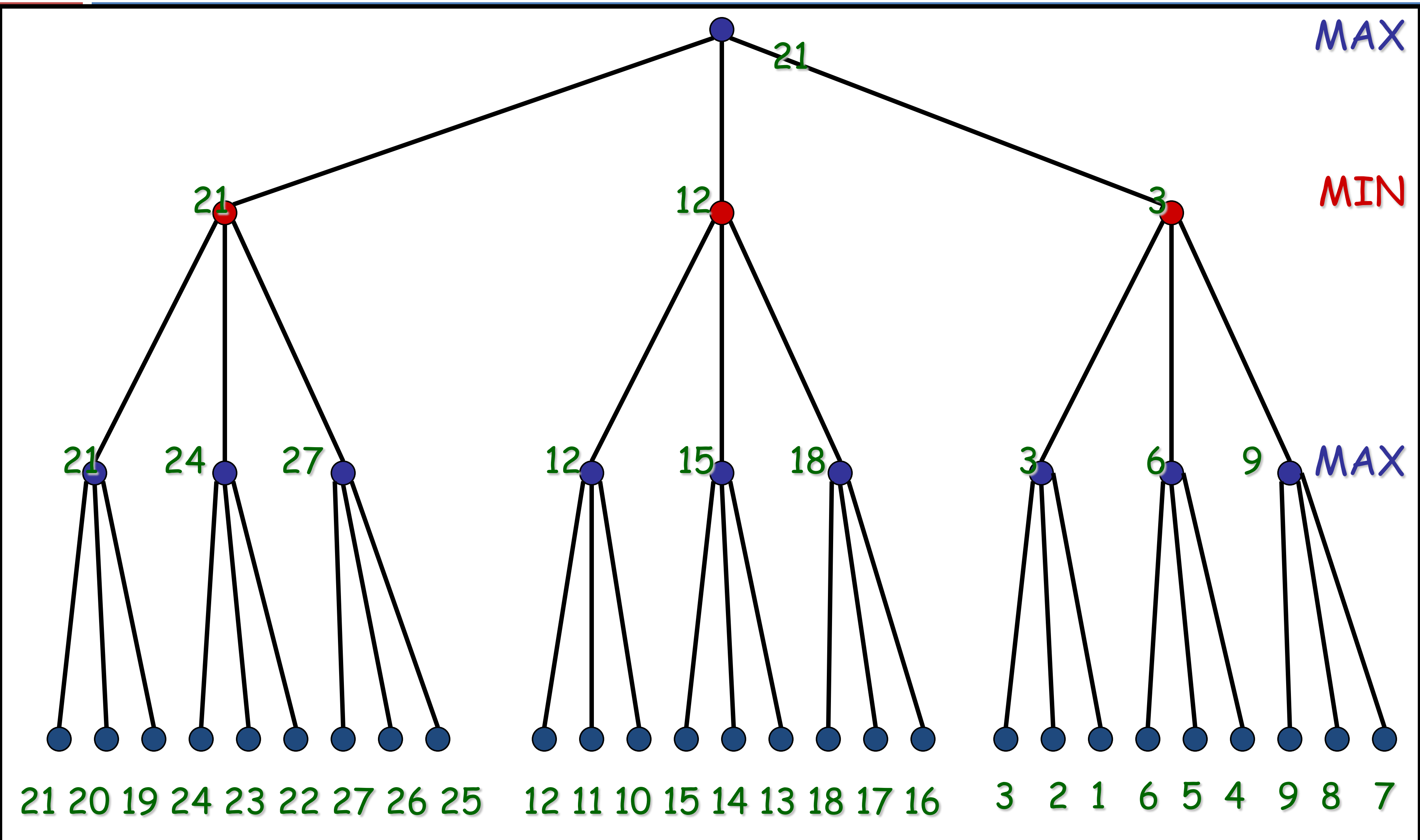


The Gain: Best Case:

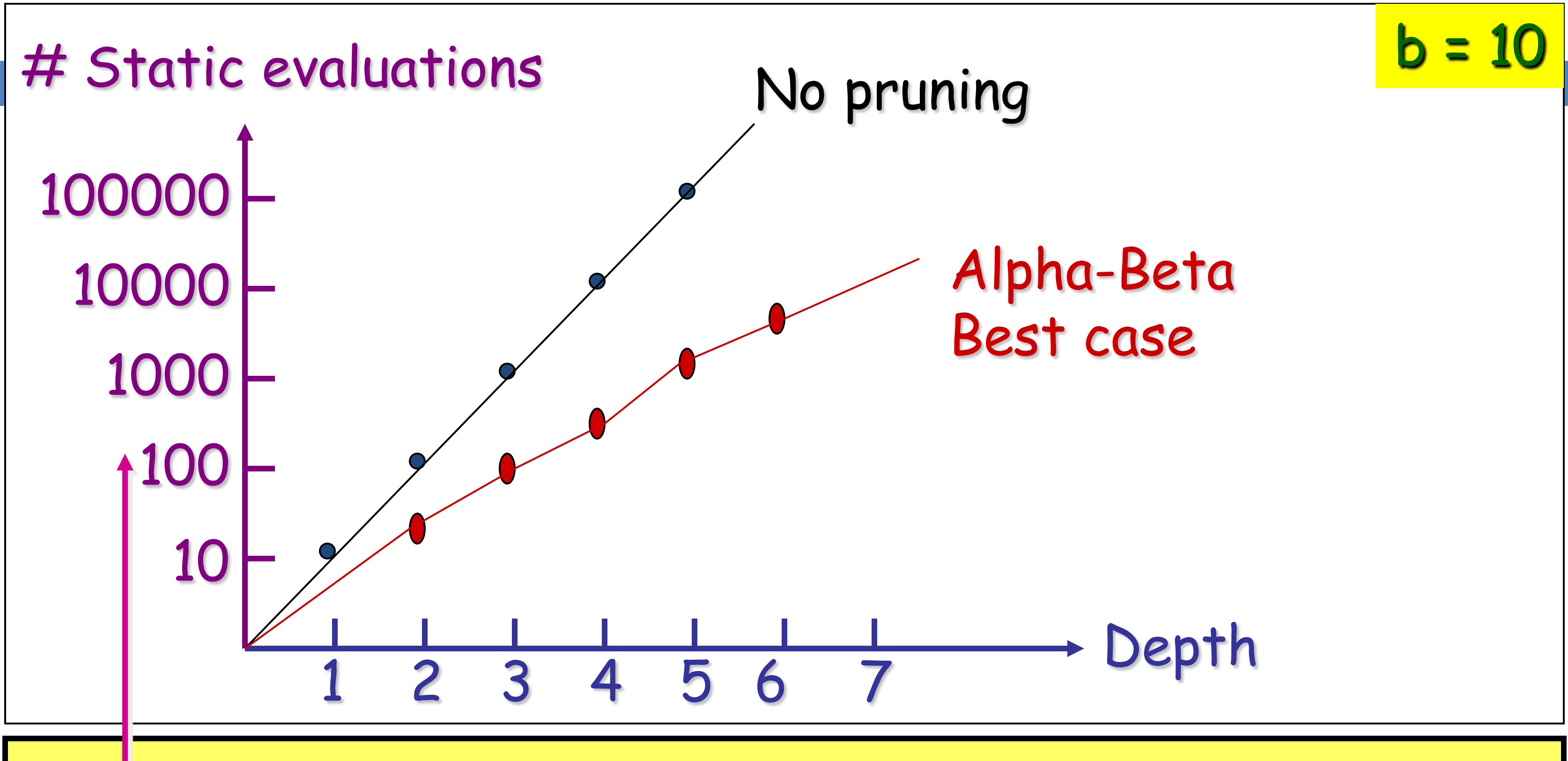
- If at every layer: the best node is the left-most one



Example of a perfectly ordered tree



Best case gain pictured:

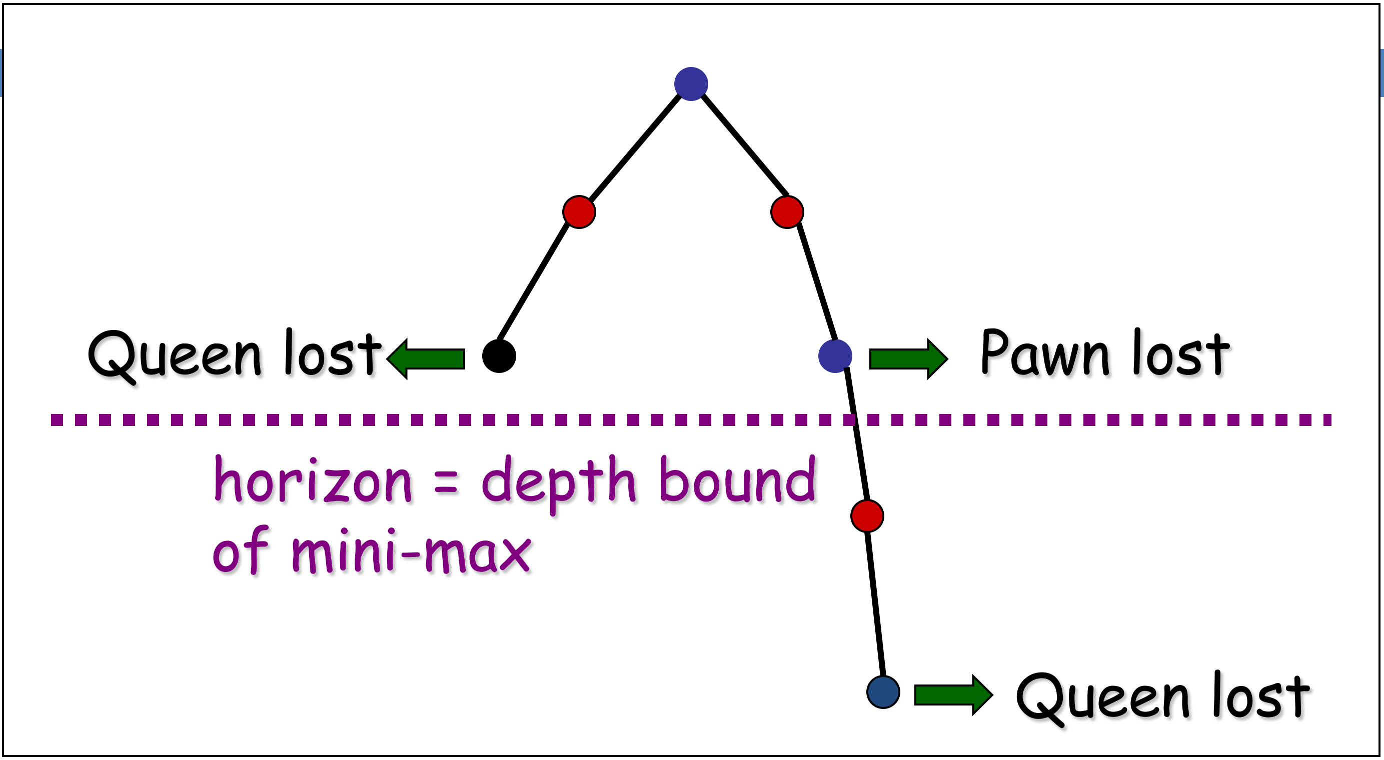


- Note: algorithmic scale.
- Conclusion: still exponential growth !!
- Worst case??

  For some trees alpha-beta does nothing,

  For some trees: impossible to reorder to avoid cut-offs

The horizon effect



Because of the depth-bound
we prefer to delay disasters, although we don't prevent them !!

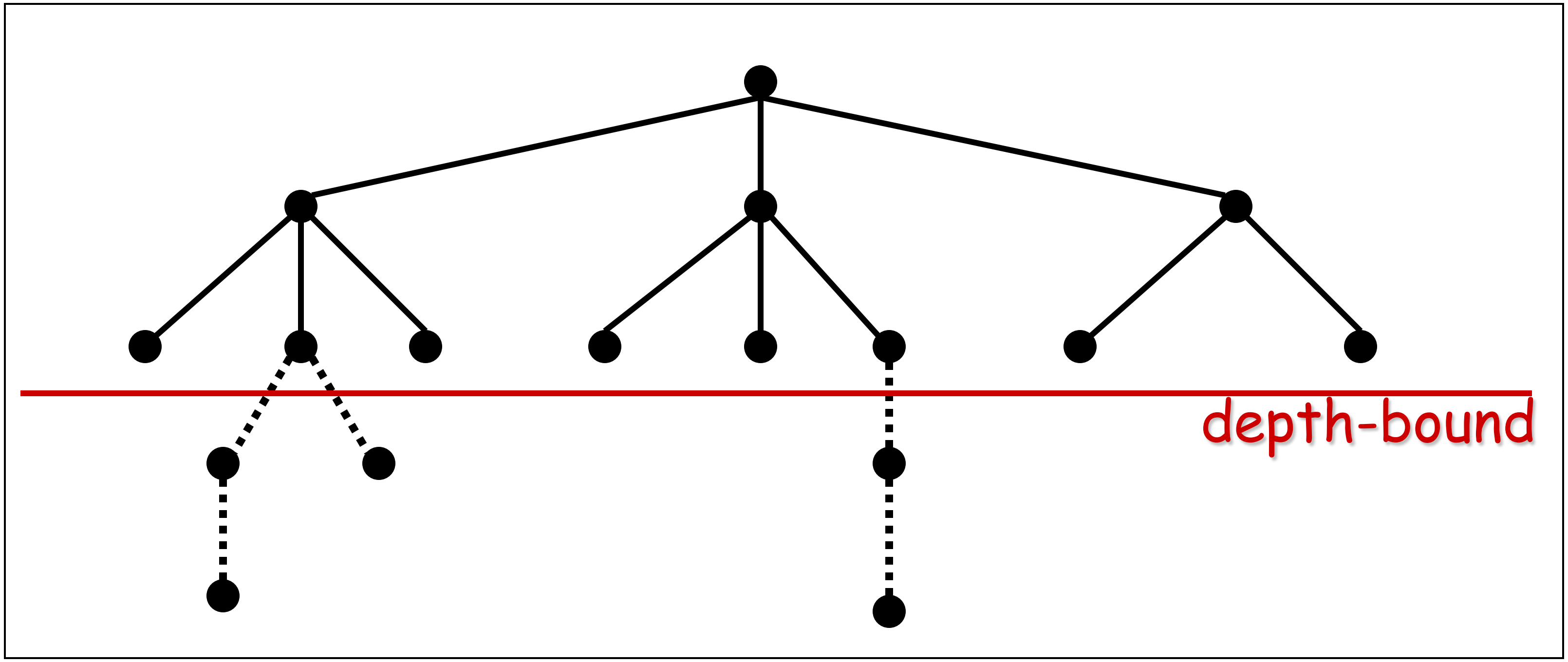
- solution: heuristic continuations

Heuristic Continuation

In situations that are identifies as strategically crucial

e.g: king in danger, imminent piece loss, pawn to become as queens, ...

extend the search beyond the depth-bound !



Time bounds:

How to play within reasonable time bounds?

Even with fixed depth-bound, times can vary strongly!

Solution:

Iterative Deepening !!!

Summary

- alpha-beta algorithm does the same calculation as minimax, and is more efficient since it prunes irrelevant branches
- usually, the complete game tree is not expanded, search is cut off at some point and an evaluation function calculated to estimate the utility of a state

- So far, for a possibly good and efficient search:
  - select good search method/technique
  - provide info/heuristic if possible
  - apply prune irrelevant branches